On the Use of Commodity Ethernet Technology in Exascale HPC Systems

Mariano Benito, Enrique Vallejo, Ramón Beivide
University of Cantabria
Santander, Spain
{mariano.benito, enrique.vallejo, ramon.beivide}@unican.es

***Abstract*—Exascale systems will require large networks with hundreds of thousands of endpoints. Ethernet technology is employed in a significant fraction of the Top500 systems, and will remain as a cost-effective alternative for HPC interconnection. However, its current design is not scalable to Exascale systems. Different solutions have been proposed for scalable Ethernet fabrics for data center, but not specifically for HPC applications.**

This work identifies the major differences in network requirements from both environments. Based on them, it studies the application of Ethernet to Exascale HPC systems, considering the topology, routing, forwarding table management, and address assignment, with a focus on performance and power.

Our scalability solution relies on OpenFlow switches to implement hierarchical MAC addressing with the introduction of compaction mechanisms for TCAM table reduction. To simplify deployment, a protocol denoted DMP performs automated address assignment without interfering with layer-2 service announcement protocols. An analysis of latency requirements of HPC applications shows that their communication phases are very short, making controller-centric adaptive routing unfeasible. We introduce Conditional OpenFlow rules as an instrument which allows for adaptive routing with proactive rule instantiation. Routing decisions are taken in the switch depending on network status, without controller interaction. This mechanism supports multiple topologies which require minimal or non-minimal adaptive routing and improve performance and power.

Altogether, this work introduces a realistic and competitive implementation of a scalable lossless Ethernet network for Exascale-level HPC environments, considering low-diameter and low-power topologies such as Flattened Butterflies or Dragonflies, and allowing for power savings up to 54%.

***Index Terms*—Exascale HPC; Interconnection Network; SDN**

I. Introduction

Technology evolution has led to a convergence in Data Center (DC) and High-Performance Computing (HPC) systems. In fact, similarly to the introduction of commodity x86 processors in HPC systems in the 90's - 2000's, nowadays a significant part of the HPC systems rely on commodity Ethernet technology. Figure 1 shows the evolution of the system-level interconnection technology employed by supercomputers in the Top500 ranking [1]. From its introduction in HPC systems in the early 2000's, Ethernet technology has been used in a significant fraction of the systems.

Ethernet's large economy of scale [2], the advent of simple whitebox switches [3] based on merchant silicon, the possibility of lossless implementations [4], and the ubiquity of Ethernet NICs in SoCs or motherboards suggests it will remain as a cost-effective alternative for HPC interconnection.

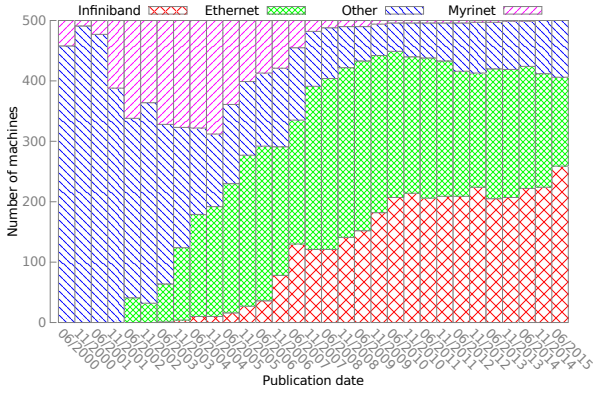


Fig. 1. Number of systems in the Top500 list by interconnection family (plot in order by current % of total) from 2000's.

For example, the Mont-Blanc project [5] aims to design an Exascale-level HPC system based on commodity SoCs used in embedded systems, leveraging on their low power and high computing efficiency [6]. Such embedded chips often implement an Ethernet NIC, and in fact the Mont-Blanc prototype interconnection relies on commodity Ethernet technology.

However, such simple interconnect design is not scalable to Exascale systems. The number of computing nodes required for such system would be very large; with 3 TeraFLOPS nodes, more than 300,000 of them would be required, clearly exceeding the capacity of the forwarding tables of any commodity switch. Scaling these tables would impact both switch latency and power consumption; indeed, alternative low-diameter topologies have been proposed for large-level HPC systems which help reduce the overall power consumption.

Although several techonologies have emerged to scale Ethernet networks, such as overlay encapsulations or Software Defined Networking (SDN) solutions based on OpenFlow [7], they are not necessarily suited for an HPC environment. In fact, the flexibility and migrability requirements in a DC differ from the low-latency and high-throughput goals in HPC systems. Based on an analysis of requirements of HPC applications, this paper identifies the shortcomings of large-scale Ethernet deployments and introduces efficient alternatives for Exascale-level HPC environments. This proposal constitutes a novel practicable implementation of HPC networks based on commodity Ethernet switches, including the use of low-

diameter topologies such as Flattened Butterflies [8] (FBs) or Dragonflies [9] (DFs). In particular, the main contributions of this paper are the following:

- A comparative study of the different requirements of HPC and DC networks, highlighting the need for proactive flow rules and non-minimal adaptive routing.
- An analysis of the suitability of Ethernet scalability mechanisms in HPC environments, introducing techniques for topology-dependant forwarding table compaction.
- The Dynamic MAC Protocol (DMP), a mechanism to dynamically identify the hosts based on rewritten media access control (MAC) addresses.
- *Conditional Flow Rules* for minimal and non-minimal adaptive routing in OpenFlow-based *direct* topologies, which allow to implement low-diameter networks with up to 54% power savings.

The remainder of this paper is organized as follows. Section II analyzes the different requirements in HPC and DC environments, highlighting power consumption issues. Scalability solutions for Ethernet networks are discussed in Section III. Section IV studies the associated service discovery and MAC address rewriting interferences and introduces a dynamic MAC address assignment protocol. Multipath adaptive routing in different topologies considering HPC application traffic is discussed in Section V. Section VI presents the evaluation environment and the results of tests and simulations. Finally, Section VII summarizes the related work and Section VIII concludes the paper.

II. HPC interconnection requirements and problem statement

Scalable SDN solutions have been proposed for large-scale DC networks based on commodity OpenFlow Ethernet switches. However, network requirements and traffic characteristics in an HPC environment differ from those in traditional DC, making the optimal solutions in one case not so suited to the other. In particular, the main requirements identified to differ in HPC from traditional DCs are:

- Low latency is crucial for application performance, making controller-based flow instantiation unfeasible.
- The communication stack is not necessarily TCP/IP.
- Topologies other than traditional Folded-Clos have been proposed specifically for large HPC systems, with a special focus on both scalability and power saving.
- HPC communication phases are typically shorter.

The next paragraphs analyze these differences and map them to implications on the underlying network fabric.

Low latency requirements translate into the need of proactive forwarding rules [10] (rather than reactive ones instantiated when a flow begins) to avoid traffic indirections in the critical path. An SDN controller should learn the network topology and set the forwarding rules in advance, like the Subnet Manager in Infiniband [11] networks. Of course, network flooding due to missing forwarding entries (used in traditional Ethernet with conversational learning, 802.1D [12]) should be avoided.

HPC alternatives to the TCP/IP communication stack, such as Open-MX [13] and RDMA over Converged Ethernet [14] (RoCE), do not employ IP addresses nor TCP ports. This makes a single layer-2 domain compulsory. A traditional Ethernet design with flat addressing in an Exascale system would require CAM forwarding tables with hundreds of thousands of MAC entries. That is impractical with traditional switch hardware (which typically ranges from 4K to 64K entries), not only because of table capacity limitations, but also of power and latency concerns. Models of power consumption of Content-Addressable Memory [15] (CAM) or Ternary Content-Addressable Memory [16] (TCAM) tables shows power scales roughly proportionally with table size. In particular, the main power consumption in TCAMs comes from its *matchline* logic, which grows linearly with the number of entries. The consumption of these tables typically reaches several tens of Watts [16], [17], being the most hungry modules besides ports SerDes [18].

DC interconnection networks traditionally rely on some form of tree or Folded-Clos topology. Implementation costs and energy consumption restrictions suggest the use of alternative, direct topologies with low diameter, such as Flattened Butterflies or Dragonflies. Compared to Folded-Clos, these topologies employ a lower number of switches and links for a given network size, leading to a lower power consumption in switches logic and link SerDes, and lower installation cost. However, minimal paths in these topologies are heavily congestion-prone, requiring *non-minimal* adaptive routing.

Finally, communication phases are much shorter in HPC applications than in DC workloads (especially in DC user-facing applications). This makes controller-based per-flow traffic engineering (adaptive routing) unfeasible.

III. Scalability mechanisms in Ethernet networks

Traditional Ethernet employs flat routing based on the hosts MAC addresses. Switches' CAM forwarding tables employ an entry for each endpoint in the network. Conversational MAC address learning in traditional Ethernet switches [12] fills these tables dynamically when network conversations start, avoiding entries for destinations without communication. However, this reactive mechanism implies that frames destined to unknown entries are flooded across the whole network to make sure they reach their destination. Indeed, the major disadvantage of a flat addressing is the required size of the switches' MAC address table; when it overflows, traffic to MACs in excess are flooded as if they were unknown destinations.

Overlay mechanisms have been proposed to build large layer-2 fabrics. TRILL [19] and FabricPath [20] employ MAC-in-MAC encapsulations to reduce the forwarding table use. In these hierarchical implementations, ingress switches encapsulate the original Ethernet frame in a new frame using the destination switch ID for the destination MAC field; egress switches remove the outer header to recover the original frame. Switches in the *core* of these networks only need to hold CAM entries for the network switches, not the network hosts. By contrast, *access* switches still require an entry for each host

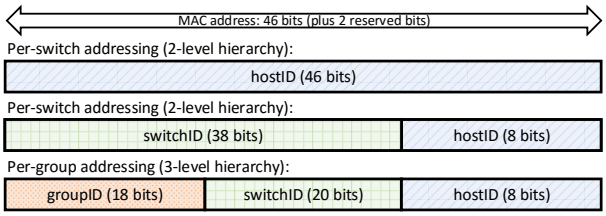


Fig. 2. Hierarchical addressing alternatives.

involved in a conversation. This could explode to the whole network size in the worst case. Conversational learning is also employed by default in these proposals, what implies that unknown (or overflown) entries are flooded. VXLAN [21], which is a MAC-over-UDP/IP overlay encapsulation, also relies on conversational learning and does not reduce the maximum size of access switch tables.

The previous approaches rely on searches on CAM tables for a match of the complete MAC address. By contrast, OpenFlow switches employ TCAM tables allowing for partial matches and hierarchical MAC organizations. PortLand [22] and MOOSE [23] introduce layer-2 hierarchical routing, by organizing hosts in groups sharing a common MAC prefix and setting a single routing rule for all hosts in the same destination group. A unique pseudo-MAC (PMAC) address is assigned to each host, encoding its location in the network. Access switches dynamically modify the MAC field of the frames received from or sent to its own nodes (MAC to PMAC and PMAC to MAC), so the network only detects PMACs and destination hosts maintain the illusion of unmodified MAC.

Our scalability solution will rely on hierarchical MAC addresses, since they require a limited number of proactive forwarding rules and do not flood traffic. The number of flow entries depends on the definition of groups. Subsection III-A analyzes the rules required for different address hierarchies and topologies, and Subsection III-B studies rule compaction.

A. Scalability analysis of hierarchical addressing

The minimum size for an address block corresponds to the set of hosts connected to the same switch (similar to TRILL and FabricPath), because smaller partitions do not provide any reduction in rules. In addition, larger groups can be formed by addressing several switches with a common group prefix, and forwarding traffic according to such group prefix. This three-level hierarchy (host, switch and group) reduces the number of flow rules because a single rule is required for remote groups. Figure 2 depicts these options; the specific number of bits of each field may vary according to the controller policy.

The number of flow rules required is discussed next, using the following notation: G is the number of groups ($G = 1$ for a two-level hierarchy); S is the number of access switches (with directly-connected hosts) per group; and T is the number of hosts on each access switch. The number of rules R required in each access switch is:

$$R = (G - 1) + (S - 1) + T \quad (1)$$

TABLE I
NUMBER OF PORTS DEDICATED TO HOSTS AND NETWORK SCALABILITY IN DIFFERENT TOPOLOGIES, USING SWITCHES WITH P PORTS.

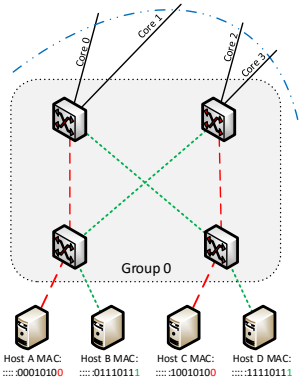
Topology	Hosts per access switch, T	Hosts per network switch	Scalability (max hosts, H)
3-Level Folded-Clos	$\approx P/2$	$\approx P/5$	$P^3/4$
2D Flattened Butterfly	$\approx P/3$	$\approx P/3$	$\approx (P/3)^3$
3D Flattened Butterfly	$\approx P/4$	$\approx P/4$	$\approx (P/4)^4$
Dragonfly	$\approx P/4$	$\approx P/4$	$\approx 4 \times (P/4)^4$
4D Flattened Butterfly	$\approx P/5$	$\approx P/5$	$\approx (P/5)^5$

The network size will be determined by the topology and switch size. The proportion of switch ports used to connect hosts depends on the topology. Table I represents this value for several topologies: 3-level Folded-Clos, Flattened Butterflies and Dragonflies. Folded-Clos are *indirect* topologies with transit switches to which no host directly connects, so the overall proportion of host ports in the network is lower than in access switches. Flattened Butterflies and Dragonflies are recent *direct* topologies which are used in commercial HPC systems. Since *direct* topologies do not have transit switches, the proportion is the same in both cases. The values of the table roughly determine the number T of hosts on each switch, for a given switch size. The number of hosts is given by $H = G \times S \times T$, with $G = 1$ for the two-level hierarchy.

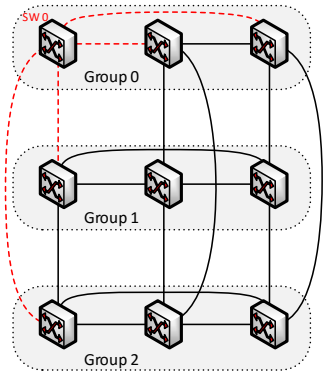
The minimum number of entries for a given size is obtained in Equation 1 with $G \approx S$. However, the number of switches assigned to each group might depend on the network topology; Figure 3 represents the three topologies considered. In the three-level Folded-Clos in Figure 3a, nodes are organized in several 'pods' which are connected via core switches with twice as many 'pods' as access switches per pod. Its natural division is one group per 'pod'. The 2D FB in Figure 3b has as many rows as columns, with an all-to-all connection per row and column. Matching a row to a group leaves $S = G$. With more dimensions, the same assignment of one group per row leaves $G > S$. Finally, the DF topology in Figure 3c is naturally organized in groups, but their amount is significantly larger than the number of switches per group.

Considering this organization, Figure 4 depicts the number of flow rules required to support a given topology and size, considering both flat, 2-level and 3-level hierarchical address organizations. 'Flat addressing' represents the traditional CAM-based implementation with one entry per host and it is included for comparison. The scalability (number of hosts in the horizontal axis) only depends on the switch size and topology; the number of flow rules (vertical axis) also depends on the use of per-switch or per-group addressing.

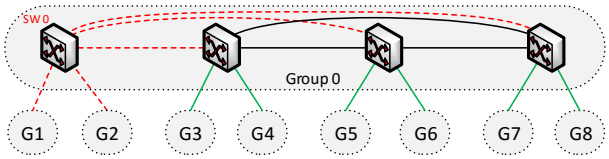
The size of TCAM tables in commodity Ethernet switches usually ranges from $4K$ to $32K$ entries [24]. As observed in the plot, using 3-level hierarchical addressing and DF or 3D FB, it is possible to reach more than the 300,000 estimated hosts for future Exascale HPC systems using a layer-2 network fabric requiring less than 4K rules.



(a) 3 level Folded-Clos with k=4 ports routers. Only one 'pod' is shown, the third upper level and the rest of pods are omitted for simplicity.



(b) 2D Flattened Butterfly highlighting one switch and their links. Each row corresponds to a group.



(c) Dragonfly with 9 groups, $G0 - G8$. Only group $G0$ is detailed.

Fig. 3. Different topologies being considered.

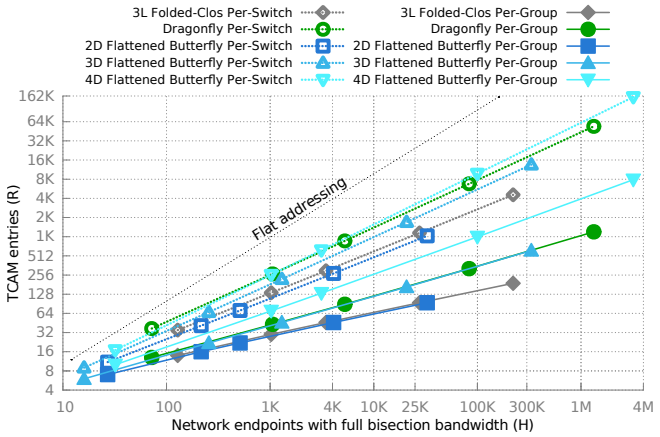


Fig. 4. Number of TCAM entries required for varying network size and topology, using per-switch or per-group addressing. The points correspond to the maximum network size using switches with 8, 16, 24, 48 or 96 ports.

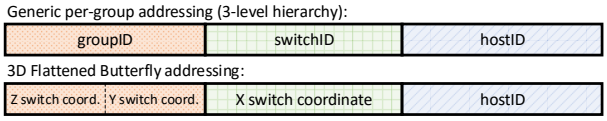


Fig. 5. Hierarchical addressing in 3D Flattened Butterflies considering forwarding rule compaction.

B. *Scalability analysis with TCAM rules compaction*

This subsection explores the use of wildcards for table compaction for the three previous topologies. Equation 1 considered one flow rule for each possible destination (host, switch or group). However, when several destinations share the same output link, they might be merged into a single rule depending on their addresses.

The minimum number of TCAM entries in a switch equals its port count, since at least one entry is required to output frames on each port. This lower bound is reached naturally in the 2D FB topology with per-group addressing. This occurs because each switch is directly linked to its own T endhosts, the $(S-1)$ other switches in its own group (different columns) and the remaining $(G-1)$ groups (different rows) without overlap. This is highlighted in Figure 3b for switch 'SW 0'. For three or more dimensions, a similar reduction is feasible, by simply assigning the switch location coordinates (X, Y, Z) to different sub-fields in the header as presented in Figure 5 and using wildcards in the corresponding sub-fields. Thus, a FB of any dimension can reach the minimum number of rules.

Figure 3a presents an excerpt of a Folded-Clos, showing one 'pod' and the uplinks to core routers (level three of the network). Routing in the Folded-Clos is *up-down*. The uplink followed to core routers is independent of the destination, since all core routers are ancestors of all pods (the same applies to neighbor switches in the pod). Thus, a hash of the source fields can be used for static balancing of the uplinks, as in the example presented in the figure. Again this leads to a number of rules equal to the switch ports.

On the contrary, in the DF topology it is not possible to reduce the number of rules to the number of switch ports. To illustrate it, consider the leftmost switch in Group 0 in Figure 3c. It requires one rule for each of its T hosts, one rule for each of the $S-1$ direct neighbours in the group, and one rule for each directly connected remote group, which equals the port count. However, it also requires additional rules to reach the groups connected to other switches in its group. At least, this would imply $S-1$ additional rules if all the rules associated to a given output port could be compacted. However, in the general case this is not possible because the addresses of the remote groups are not necessarily aligned, as in the example in the cited figure: using wildcards to compact rules for remote groups 1 and 2 would also match group 3.

Figure 6 shows the number of rules for the topologies considered. In the case of the DF, the maximum and minimum number of rules is presented; the actual number will be in-between and will vary from switch to switch. In the other cases, the lower limit is reached. Even in the worst case of the DF with more than 1 million nodes, the number of rules is relatively low and fits in existent OpenFlow switches.

IV. Service Announcement and MAC Address Rewriting

This section identifies a problem generated by dynamic rewriting of MAC addresses and introduces an alternative mechanism to automate host MAC configuration. The scalability solution in Section III requires rewriting MAC addresses to include location information. Previous proposals implement dynamic rewriting of the frame headers in access switches

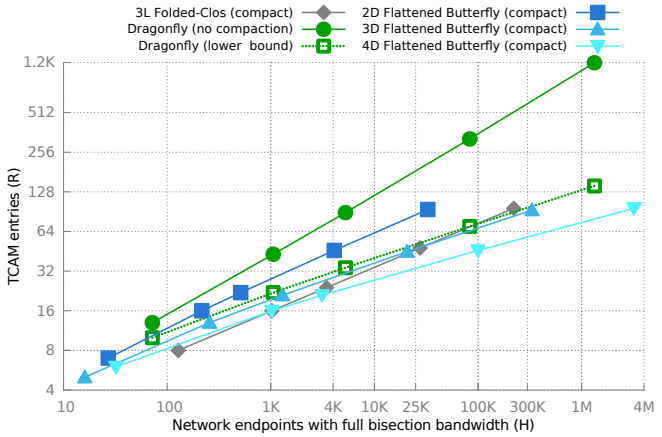


Fig. 6. Number of TCAM entries after compaction. The points correspond to the maximum network size using switches with 8, 16, 24, 48 or 96 ports.

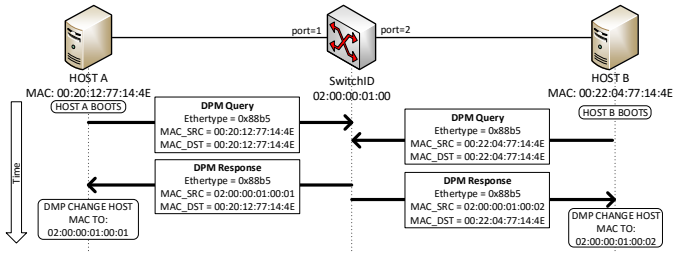


Fig. 7. Sequence of packets when two hosts boot using DMP.

[22], [23]. However, this interferes with layer-2 service announcement mechanisms, used for example in Open-MX.

Distributed discovery mechanisms rely on requests from the service users (such as ARP for IP-to-MAC mapping) or announcements from the service provider (e.g. Open-MX computing nodes announcing their availability). In the first case, requests can be intercepted by the access switches and derived to a central fabric manager, as done in PortLand [22] and SEATTLE [25] to attend ARP requests. In the second case, by contrast, broadcast and multicast messages announce the service to all nodes so a similar centralized solution is not feasible. Furthermore, MAC rewriting interferes with these announcements. In fact, Open-MX announcements include the source node MAC address in the data field, so receivers of this frame record the original, unmodified address and communication never happens.

Two alternative solutions are introduced next. The first one modifies the service discovery protocol, with receivers recording the MAC address in the announcement frame header, rather than the data field. This is compatible with on-the-fly address rewriting and is possible in Open-MX because it is open-source, but it might be unfeasible with proprietary stacks.

The second alternative modifies the MAC address in the hosts configuration, rather than modifying it in the frame headers. Since individually modifying each host's address is unfeasible with hundreds of thousands of nodes, a control protocol has been designed to automatically configure the host MAC at boot time. This protocol has been denoted Dynamic MAC Protocol (DMP) and is depicted in Figure 7. When a host boots, it asks for a MAC address by sending a DMP request using a specific EtherType value. Its own switch identifies this query and sends the frame back, overwriting the source address field with the new hierarchical MAC address, as discussed in Section III. One rule per connected host is required in addition to the routing rules discussed in the previous section. This approach does not restrict the use of the host MAC address in the announcement payload nor the use of hierarchical routing.

Both alternatives have been implemented and verified in a computer using Open vSwitch and 4 virtual machines. In these tests OpenFlow 1.0.0 [26] has been used with the optional feature "Modify-Field" to change the header MAC field. The EtherType value 0x88b5 reserved for experimental purposes has been used on DMP packets in the tests.

V. Multipath Adaptive Routing

This section explores adaptive routing in Ethernet networks. Oblivious multipath routing employs a fixed function for load balancing, typically a hash of some header fields. Such approach is used in traditional link aggregation (LAG) or equal-cost multipath (ECMP) routing. Its disadvantage is that the paths might receive a different load, leading to suboptimal throughput. By contrast, adaptive routing dynamically balances the load of each path. An adaptive routing implementation is more complex since it requires an estimation of the network load and instantiating additional routing entries.

An initial analysis in subsection V-A will conclude that per-flow load estimation latency is too large for typical HPC applications. Section V-B introduces *conditional* OpenFlow rules triggered by flow-control messages. This is applied to both ECMP in subsection V-C and to non-minimal routing in subsection V-D. A discussion is presented in subsection V-E.

A. Latency requirements for adaptive routing in HPC and controller-based estimation of per-flow offered load

Several traces of applications from the NAS Parallel Benchmarks [27] have been analyzed, from runs with 64 MPI processes using 64 nodes of a cluster. Figure 8 shows a visualization of four iterations of the CG kernel. Blue sections represent computation phases, orange sections represent communication ones and yellow lines are point-to-point messages. Four iterations of the algorithm last 6.48 ms (the timescale is indicated in the bottom of the figure in small font), resulting in changes of traffic in less than 2 ms. A similar visual analysis of other benchmark applications gives rise to the values in Table II. Iterations range from 2 to 58 ms, meaning that traffic changes are even quicker. By contrast, typical DC applications can suffer from congestion periods lasting for several seconds [28].

Several mechanisms rely on controller-based estimations of per-flow offered load in order to adapt routing [29]–[31]. However, their reaction time exceeds 70 ms in the best case, due to monitoring intervals and remote controller communication. Planck [32] captures traffic samples to react in the order of

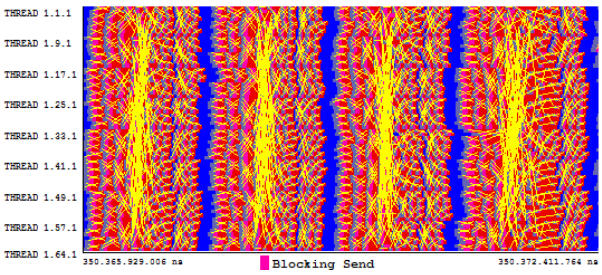


Fig. 8. Visualization of CG using 64 MPI processes. The timeframe shown comprises 6.48 ms for 4 iterations.

TABLE II
APPROXIMATE ITERATION TIME OF NPB APPLICATIONS.

Application/kernel	Iteration time
CG	1.6 ms
BT	29 ms
FT	58 ms
IS	10 ms
LU	52 ms
MG	11 ms
SP	22 ms

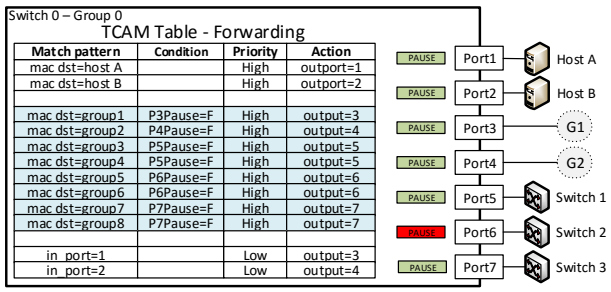


Fig. 9. Switch architecture with conditional flow rules. When the condition fails (when the output port is paused) the high-priority minimal routing rule is ignored, leading to the use of a low-priority rule.

few milliseconds. However, it requires a significant monitoring infrastructure and relies on TCP sequence numbers to estimate flow rates, making it specific for traditional DC environments. As far as we know, no other monitoring mechanisms achieve reaction times similar to Planck, making controller-centric traffic engineering unfeasible in HPC environments. Therefore, reactive flow rules instantiated by the controller are not well suited to adapt switch routing to changing HPC traffic.

B. Proactive conditional OpenFlow rules

To solve the aforementioned problem, we propose the idea of conditional flow rules. These rules are proactively instantiated by the controller, and they are deactivated locally by the switch on network events, such as link-level flow control pauses. Using this idea, traffic is diverted to alternative paths when a preferred one gets blocked due to congestion.

Figure 9 depicts the idea of conditional flow rules with the required changes to the flow table. Besides ordinary rules, the system relies on two sets of rules, *default* and *alternative*, with different priority. Both ordinary and *default* rules correspond to the ones determined in Section III for hierarchical routing, and have high priority. *Default* rules are conditional, dependant on the pause status of their associated output port (highlighted in the figure). *Alternative* rules employ lower priority. Commonly, both *default* rules and their corresponding *alternative* backup rules match for a given frame, but *default* rules are used because of their higher priority. However, when a given output port receives a PAUSE, the corresponding *default* rules are deactivated, so *alternative* rules start to be used, diverting traffic.

With this proposal, default paths are used until their queues become completely full. Both *default* and *alternative* rules are proactively instantiated by the controller after topology discovery. Their application for minimal or non-minimal multipath routing is discussed in the next subsections.

C. Conditional OpenFlow rules for minimal routing

Conditional OpenFlow rules can be applied to topologies with multiple minimal paths by assigning a *default* rule to each of the default paths, and one or more *alternative* rules (for each alternative minimal path, with different levels of low priority) to paths already assigned to other *default* rules.

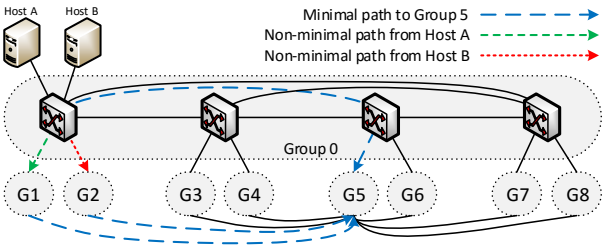
In the case of the Folded-Clos with compact routing rules, one *default* rule is assigned to each uplink using per-source hashing (per-destination hashing is equally possible). Up to $k-1$ additional *alternative* rules can be assigned to the same per-source hash, for the $k-1$ remaining uplinks. This multiplies the number of uplink rules in a factor up to k.

In N-dimensional FBs there exist up to N potential outputs for a given destination, for the N options for the first hop. Interestingly, in this case multipath routing can be implemented without increasing the number of rules with respect to the compact case. Higher-level rules (e.g., to reach a remote group) will be conditional with high priority, while lower-level rules (e.g., destination switch index) will have low priority and employ wildcards on the group bits.

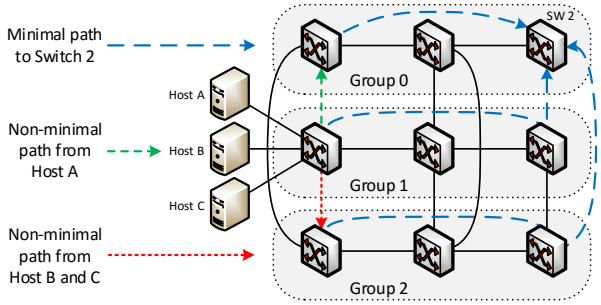
D. Conditional OpenFlow rules for non-minimal routing

Several *direct* topologies proposed for HPC environments can suffer heavy congestion under adversarial traffic patterns. In a DF, when all the nodes in a group communicate with nodes in another group, the global link between them needs to pass all the traffic and quickly saturates. Since there is no path diversity in a Dragonfly, the options introduced in the previous subsection do not apply. The same occurs in a FB using Dimension-Ordered Routing when all end hosts connected to a switch send traffic to the same destination switch. In such cases, *non-minimal* routing can balance traffic by using longer paths. Valiant routing [33] selects a random intermediate switch in the network, sends traffic to this switch and then to the final destination. Adaptive non-minimal routing mechanisms select between minimal or Valiant routing in these networks [8], [9], [34].

Non-minimal routing using conditional OpenFlow rules relies on diverting traffic to a given intermediate destination when



(a) Dragonfly



(b) Flattened Butterfly

Fig. 10. Non-minimal routing in a Dragonfly (a) and Flattened Butterfly (b) networks using conditional OpenFlow rules.

the corresponding minimal path is congested. The intermediate destination cannot be selected randomly without intervention from the controller, so a given Valiant intermediate switch will be statically assigned to each source node.

Figure 10a shows the application of conditional *non-minimal* routing to DFs. In this topology, global links are the most congestion-prone ones. Each switch will employ several high-priority rules for minimal routing as discussed in Section III-B. Of these, rules for remote groups will be conditional. T additional low-priority rules are included to forward traffic from injection ports to remote groups (note that the input port can be checked in OpenFlow rules). When the conditional minimal rule is deactivated, these additional rules forward traffic directly to global links, towards a remote intermediate group. From that point, minimal routing is employed. Since a balanced DF has as many end hosts per switch (T) as global ports [9], one non-minimal output can be assigned to each endhost, effectively balancing traffic. This mechanism requires T additional rules.

The application to FBs is similar, as depicted in Figure 10b. In this case, each switch is connected to $S-1$ remote groups but there are $T = S$ hosts per switch to archive a balanced design [8], so one of the hosts will not have a non-minimal path assigned (or one of them will be repeated).

E. Discussion

The use of topologies with cycles together with lossless link-level flow control introduces deadlock issues in the network, which need to be handled in pair with routing. For Folded-Clos, simple Up-Down routing avoids deadlock issues. Several custom proposals for HPC rely on multiple Virtual Channels (VCs) traversed in ascending order, based on a previous work [35]. In particular, N VCs are required for fully adaptive routing in $N-$dimensional FBs (typically $N = 2$ or $N = 3$) and just 3 VCs are required for DFs (which can be reduced to 2 in our adaptive implementation, since there are at most 2 local hops and 2 global hops). These VCs can be mapped directly to two different Ethernet Class-of-Service (CoS) levels and different switch buffers, leaving enough CoS levels to differentiate other types of traffic. CoS updates can be embedded in existent OpenFlow rules.

The proposed adaptive routing decision relies on snooping flow control messages. Alternative implementations might rely on explicit congestion notifications, such as in IEEE 802.1Qau.

Adaptive routing mechanisms can increase traffic throughput at the cost of out-of-order delivery. It is the responsibility of the transport protocol to detect and reorder network traffic in such cases. We do not focus on such protocols in this paper.

The use of a statically pre-selected Valiant path differs from the original random definition. Additional non-minimal paths (with varying levels of low priority) can be included, at the cost of an increased number of rules.

VI. Evaluation

This section evaluates the network power consumption for different topologies and TCAM organizations and evaluates the performance of conditional Openflow rules.

A. Topology power comparison

We compare the power consumption of the three topologies analyzed across the paper (Folded-Clos, FB, DF). We consider a network supporting $\approx 300,000$ hosts with full bisection bandwidth, built using 72 port Ethernet switches. Power calculations consider 40 Gbps ports and worst-case 64-byte packets. The energy required for reading and writing a 64-byte packet from the buffer memories is 4.5048 nJ and 4.4993 nJ respectively, following the calculations in a previous work [18]. Optical ports consume ≈ 0.6 Watts (4 lanes of 150 mW each) and the electrical ones, used to connect hosts in access switches, 20% less [36]. A fixed value of 30 Watts has been considered for the CPU and logic. The power consumption for OpenFlow 1.5 TCAMs (which can match more than 1,000 header bits) are modeled with the tool presented in [16] considering 32 nm CMOS technology.

The number of switches differs per topology. In Folded-Clos a 4-stage topology is required for the desired size. The design relies on basic 'pods' with 36 switches in the first and second levels and 1,296 hosts. 1,296 stage-3 switches connect 36 'pods' in a stage-3 group, which is replicated 6.5 times to reach 29,484 switches and 303,264 hosts. The FB requires 4 dimensions, with switches organized in a $15 \times 15 \times 15 \times 6$ array to reach 20,250 switches and 303,750 hosts. Finally, the DF employs 463 groups of 36 switches and 648 hosts, leading to 16,668 switches and 300,024 hosts. These networks do not reach the maximum size of each topology; in all cases, additional links in spare switch ports are considered to provide full bisection bandwidth.

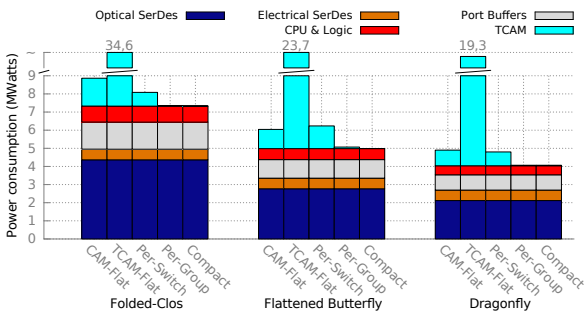
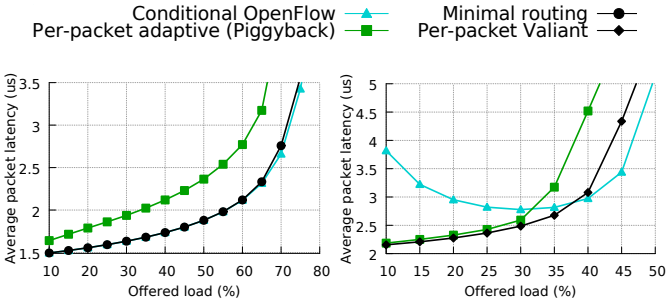


Fig. 11. Network power consumption dissection.



(a) Uniform traffic (b) Adversarial traffic

Fig. 12. Average latency under uniform (a) and adversarial (b).

TABLE III
SIMULATION PARAMETERS.

Parameter	Value
Topology	Dragonfly
Switch port count	31 ports
Link speed	40 Gbps
Packet size	1,000 bytes
Switch frequency	1 GHz
Switch latency	200 ns
Local/Global link latency	40/400 ns (8/80 m)
CoS levels used (deadlock avoidance)	2
Switches per group	16 switches
Groups	129 groups
Total end hosts	16,512 hosts

Both CAM and the four TCAM organizations from Section III are compared: 'flat' represents a traditional Ethernet with per-host addressing, 'Per-Switch' and 'Per-Group' are 2-level and 3-level hierarchical routing respectively. In the Folded-Clos, the 3-level organization considers a 'pod' to share a common group prefix. In FB, a group is the set of routers in the first dimension whereas in the DF, each group is mapped to a DF group. Finally 'Compact' is the TCAM compaction presented in Section III-B.

Figure 11 presents power results. In all topologies, the use of flat addressing with TCAMs is clearly unfeasible, since these huge tables would consume more than 900 W per switch and at least 15 MW overall. Per-switch addressing reduces these values to be competitive with the original CAM approach, but the overall consumption ranges around a MW. Per-group addressing significantly reduces TCAM power, getting very close to the optimal solution based on compaction. Once TCAM power is minimized, the impact of topology (which determines the number of switches) is what drives power consumption. The DF topology, besides not being able to fully compact TCAM flow entries, offers the best result. The TCAM-compact DF setup reduces 54.1% of the original power in the reference CAM-flat Folded-Clos, 17.0% from TCAM compaction and 37.1% from topology changes.

B. Conditional OpenFlow performance

We employ the FOGSim network simulator [37] to evaluate the performance of non-minimal adaptive routing using conditional OpenFlow rules in a power-efficient Dragonfly network. The simulated network corresponds to the parameters in Table III. The simulator mimics the behaviour of conditional OpenFlow rules as described in Subsection V-B. We feed the network with synthetic traffic, with each node injecting frames according to a Bernoulli process with variable load. Two traffic patterns have been considered: Uniform (*UN*), in which the destination of each frame is any host in the network, and Adversarial (*ADV*), in which the destination of each frame is selected randomly between all nodes in the next group. *ADV* is a traffic pattern which concentrates the traffic on the single global link between two groups, so non-minimal routing is required to obtain a proper result. We use the adaptive piggyback (PB) routing mechanism [34] as a reference; it implements per-packet adaptive routing relying on state information for every global channel in its group distributed among switches. Additionally, Minimal (MIN) and Valiant (VAL) routing are the oblivious reference routing for *UN* and *ADV* respectively.

Figure 12 shows average latency results. Under *UN*, Conditional OpenFlow rules latency resembles the reference MIN routing, since traffic distribution does not generate congestion which triggers non-minimal routing. In PB part of the traffic is sent non-minimally which increases latency. In contrast, under *ADV*, OpenFlow average latency under small loads is larger than VAL. This occurs because Conditional OpenFlow rules rely on flow-level pauses, which implies that minimal paths need to be saturated for non-minimal routing to occur. The amount of traffic routed minimally experiences larger congestion and increases average latency. Still, the average latency of the proposed Conditional OpenFlow mechanism is competitive against optimized custom HPC routing alternatives.

VII. RELATED WORK

This paper is focused on commodity Ethernet technology. An alternative vendor-independent interconnection technology frequent in HPC is Infiniband [11]. Infiniband switches provide low-latency lossless interconnection, with paths defined by a central 'Subnet Manager' analogously to OpenFlow controllers. Infiniband subnetworks are limited to 48K endpoints because logical addresses are 16 bits (with one fourth of the address space reserved for multicast), but they have no other

inherent scalability limitations. A study of adaptive routing strategies in Infiniband networks can be found in [38].

Section III has already discussed alternative mechanisms to achieve scalability with commodity Ethernet switches, such as VXLAN encapsulation [21] or layer-2 multipath overlays such as SPB [39] , TRILL [19] and vendor variants such as Brocade's VCS [40] or Cisco's FabricPath [20]. These mechanisms still require to maintain large CAM tables with as many entries as hosts in the network (in access switches), and the required encapsulation can hinder path latency.

Adaptive routing has been considered in Section V. Proposals that employ adaptive routing using SDN include Hedera [29], ElasticTree [41] or MicroTE [42]. All these proposals rely on per-flow load estimation, with an excessive latency for HPC applications as discussed in Section V. A more detailed discussion on large flow recognition is presented in [43], considering several alternatives based on packet sampling or inline datapath measurement. By contrast, HPC implementations which support packet-by-packet adaptive routing have been implemented in or proposed for multiple topologies, such as Folded Clos [44], Flattened Butterflies [8], Dragonflies [9] or SlimFlies [45]. Their selection between minimal or Valiant (non-minimal) paths typically relies on a comparison between the credits of both outputs, which is not available in Ethernet switches relying on pause flow control. Additionally, such packet-by-packet adaptive routing is typically not supported in commodity Ethernet switches since it permits packet reordering which significantly interferes with TCP fast retransmit [46]. Minkenberg *et al.* introduced in [47] the use of multiple forwarding rules in DC bridges with flat addressing, snooping congestion notification messages to switch among them. Snooping *congestion* instead of *flow control* notifications to prioritize OpenFlow conditional rules might help improve performance under adversarial conditions; a detailed study is left for future work.

Conditional OpenFlow rules were introduced in a different context in AVANT-GUARD [48]. Their conditional flow rules are triggered when a potential attack is discovered to enforce the security policy. The proposed switch datapath is similar in both cases, but our proposal needs to detect alternative triggers such as *Pause* frames. We relied on multiple CoS levels with conditional OpenFlow rules to avoid deadlock. Alternative mechanisms rely on path restrictions, such as TCP-Bolt [49].

The power consumption of TCAM has been discussed in Section III and table size minimization has been considered across all the paper. Congdon *et al* dissect the power consumption of an OpenFlow switch in [18], but they do not consider the impact of the network topology. The impact of topology has been considered in Energy-proportional datacenter networks [36] that dynamically reduce link speed to adapt to traffic load. However, they do not consider the impact of forwarding table organization. An implementation for HPC which relies on low-power Ethernet was presented in [50]. Similarly, ElasticTree [41] completely shuts down links and modifies routing to save link power.

VIII. Conclusions

In this paper we have identified the requirements of HPC interconnection networks compared to traditional Data Center networks. We have explored proposals designed for scalable DC networks using commodity switches in Exascale-level HPC interconnects. Our results suggest that most of the proposals do not properly apply to HPC systems, such as network overlay technologies (VXLAN, TRILL) which require large switch tables relying on flooding; online MAC rewriting mechanisms which interfere with layer-2 service announcement; or fine-grained per-flow load balancing mechanisms.

By contrast, our proposal relies on hierarchical routing based on location-dependent MAC addresses, TCAM rules compaction, a dynamic mechanism to assign location-dependent MAC addresses to hosts and conditional OpenFlow rules for adaptive routing. The implementation is realistic and requires minimal changes in OpenFlow switches. This set of mechanisms permit the implementation of low-power networks based on Dragonflies and Flattened Butterflies topologies using commodity Ethernet switches. Our evaluations show that performance is optimal under uniform traffic and remains competitive against ad-hoc HPC routing mechanisms under adversarial traffic, while providing energy savings up to 54%.

Acknowledgments

This work has been supported by the Spanish Ministry of Education, FPU grant FPU14/02253, the Spanish Science and Technology Commission (CICYT) under contract TIN2013-46957-C2-2-P, the European Union under Agreement DP7-ICT-2011-288777 (Mont-Blanc 1) and DP7-ICT-2013-610402 (Mont-Blanc 2), and the JSA no. 2013-119 as part of the IBM/BSC Technology Center for Supercomputing agreement. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper and Alejandro Rico for their comments on a draft of the manuscript.

References

[1] "Top500 supercomputer ranking," 2015. [Online]. Available: http://www.top500.org/

[2] B. Casemore, L. Rosenberg, R. Brothers, R. Costello, R. Mehra, P. Jirovsky, and N. Greene, "Worldwide enterprise communications and datacenter networks 2014: Top 10 predictions," IDC report, 2014.

[3] Open Compute Project Community, "Open compute project network specifications and designs," http://www.opencompute.org/wiki/Networking/SpecsAndDesigns, 2015.

[4] *IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks–Amendment 17: Priority-based Flow Control, 802.1Qbb*, IEEE Std., 2011.

[5] "Montblanc european approach towards energy efficient high performance." [Online]. Available: http://www.montblanc-project.eu/

[6] N. Rajovic, P. M. Carpenter, I. Gelado, N. Puzovic, A. Ramirez, and M. Valero, "Supercomputing with commodity CPUs: Are mobile SoCs ready for HPC?" in *Intl. Conf. on High Performance Computing, Networking, Storage and Analysis (SC)*. ACM, 2013, pp. 40:1–40:12.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[8] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: A cost-efficient topology for high-radix networks," in *International Symposium on Computer Architecture (ISCA)*, 2007, pp. 126–137.

[9] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *International Symposium on Computer Architecture (ISCA)*, 2008, pp. 77–88.

[10] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *10th ACM SIGCOMM Conference on Internet Measurement (IMC '10)*, 2010, pp. 267–280.

[11] G. F. Pfister, "An introduction to the Infiniband architecture," *High Performance Mass Storage and Parallel I/O*, vol. 42, pp. 617–632, 2001.

[12] IEEE Computer Society, *802.1D: Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges*, IEEE Computer Society Std., 1991.

[13] B. Goglin, "High-performance message-passing over generic ethernet hardware with Open-MX," *Parallel Computing*, vol. 37, no. 2, pp. 85–100, 2011.

[14] Mellanox, "RoCE in the data center," Mellanox, Tech. Rep., 2014.

[15] I. Y.-L. Hsiao, D.-H. Wang, and C.-W. Jen, "Power modeling and low-power design of content addressable memories," in *IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, vol. 4, 2001, pp. 926–929 vol. 4.

[16] B. Agrawal and T. Sherwood, "Ternary CAM power and delay model: Extensions and uses," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 5, pp. 554–564, 2008.

[17] K. Kannan and S. Banerjee, "Compact TCAM: Flow entry compaction in TCAM for power aware SDN," in *Distributed Computing and Networking*, ser. Lecture Notes in Computer Science, D. Frey, M. Raynal, S. Sarkar, R. Shyamasundar, and P. Sinha, Eds. Springer Berlin Heidelberg, 2013, vol. 7730, pp. 439–444.

[18] P. Congdon, P. Mohapatra, M. Farrens, and V. Akella, "Simultaneously reducing latency and power consumption in openflow switches," *Networking, IEEE/ACM Transactions on*, vol. 22, no. 3, pp. 1007–1020, June 2014.

[19] R. Perlman, "RBridges: transparent routing," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2, 2004, pp. 1211–1218 vol.2.

[20] Cisco, "Nexus 7000 FabricPath whitepaper version 2.0," Cisco, Tech. Rep., 2013.

[21] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, *Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks (RFC 7348)*, IETF Std., 2014.

[22] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," in *ACM SIGCOMM Conference on Data Communication*, ser. SIGCOMM '09. New York, NY, USA: ACM, 2009, pp. 39–50.

[23] M. Scott, A. Moore, and J. Crowcroft, "Addressing the scalability of ethernet with MOOSE," in *Proc. DC CAVES Workshop*, 2009.

[24] D. Kreutz, F. M. V. Ramos, P. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *CoRR*, vol. abs/1406.0440, 2014.

[25] C. Kim, M. Caesar, and J. Rexford, "SEATTLE: A scalable ethernet architecture for large enterprises," *ACM Trans. Comput. Syst.*, vol. 29, no. 1, pp. 1:1–1:35, 2011.

[26] Open Networking Foundation, "OpenFlow switch specification version 1.0," December 2009. [Online]. Available: http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf

[27] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga, "The NAS parallel benchmarks, summary and preliminary results," in *ACM/IEEE Conference on Supercomputing*, ser. Supercomputing '91. New York, NY, USA: ACM, 1991, pp. 158–165.

[28] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *9th ACM SIGCOMM Conference on Internet Measurement Conference*. ACM, 2009, pp. 202–208.

[29] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *7th USENIX Conference on Networked Systems Design and Implementation (NSDI)*. Berkeley, CA, USA: USENIX Association, 2010, pp. 19–19.

[30] J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, A. R. Curtis, and S. Banerjee, "DevoFlow: Cost-effective flow management for high performance enterprise networks," in *9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, pp. 1:1–1:6.

[31] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *ACM SIGCOMM Conference*. ACM, 2010, pp. 339–350.

[32] J. Rasley, B. Stephens, C. Dixon, E. Rozner, W. Felter, K. Agarwal, J. Carter, and R. Fonseca, "Planck: Millisecond-scale monitoring and control for commodity networks," in *ACM Conference on SIGCOMM*. ACM, 2014, pp. 407–418.

[33] L. Valiant, "A scheme for fast parallel communication," *SIAM journal on computing*, vol. 11, p. 350, 1982.

[34] N. Jiang, J. Kim, and W. J. Dally, "Indirect adaptive routing on large scale interconnection networks," in *Intl. Symp. on Computer Architecture (ISCA)*, 2009, pp. 220–231.

[35] K. Gunther, "Prevention of deadlocks in packet-switched data transport systems," *Communications, IEEE Transactions on*, vol. 29, no. 4, pp. 512 – 524, apr 1981.

[36] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *International Symposium on Computer Architecture (ISCA)*. ACM, 2010, pp. 338–347.

[37] M. García, P. Fuentes, M. Odriozola, E. Vallejo, and R. Beivide. (2014) FOGSim interconnection network simulator. [Online]. Available: http://fuentesp.github.io/fogsim/

[38] A. Daryin and A. Korzh, "Early evaluation of direct large-scale infiniband networks with adaptive routing," *Supercomputing frontiers and innovations*, vol. 1, no. 3, pp. 56–69, 2015.

[39] IEEE 802.1aq committee, *802.1aq - Standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks - Amendment 8: Shortest Path Bridging*, IEEE Std., 2012.

[40] Brocade, "Brocade VCS fabric technical architecture," Brocade Communications Systems, Inc, Tech. Rep., 2012.

[41] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "ElasticTree: Saving energy in data center networks," in *7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*. Berkeley, CA, USA: USENIX Association, 2010, pp. 17–17.

[42] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: Fine grained traffic engineering for data centers," in *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '11. New York, NY, USA: ACM, 2011, pp. 8:1–8:12.

[43] R. Krishnan, L. Yong, A. Ghanwani, N. So, and B. Khasnabish, *Mechanisms for Optimizing Link Aggregation Group (LAG) and Equal-Cost Multipath (ECMP) Component Link Utilization in Networks (RFC 7424)*, IEEE OPSAWG Std., 2015.

[44] J. Kim, W. J. Dally, and D. Abts, "Adaptive routing in high-radix clos network," in *SC '06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, 2006.

[45] M. Besta and T. Hoefler, "Slim Fly: A cost effective low-diameter network topology," in *IEEE/ACM Intl. Conf. on High Performance Computing, Networking, Storage and Analysis (SC14)*, 2014.

[46] Y. Wang, G. Lu, and X. Li, "A study of internet packet reordering," in *Information Networking. Networking Technologies for Broadband and Mobile Networks*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3090, pp. 350–359.

[47] C. Minkenberg, M. Gusat, and G. Rodriguez, "Adaptive routing in data center bridges," in *17th IEEE Symposium on High Performance Interconnects (HOTI)*, 2009, pp. 33–41.

[48] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 413–424.

[49] B. Stephens, A. Cox, A. Singla, J. Carter, C. Dixon, and W. Felter, "Practical DCB for improved data center networks," in *INFOCOM, 2014 Proceedings IEEE*, 2014, pp. 1824–1832.

[50] K. Saravanan, P. Carpenter, and A. Ramirez, "Power/performance evaluation of energy efficient ethernet (EEE) for High Performance Computing," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, 2013, pp. 205–214.